ANALYTICAL STUDY OF MAINTAINABILITY MODELS FOR QUALITY EVALUATION

Rimmi Saini

Computer Science & Engineering Department,Greater Noida Institute of Engineering & Technology, Greater Noida,U.P., 201306,India
rimmi_saini@yahoo.com

Sanjay Kumar Dubey [2]

Department of Computer Science & Engineering, Amity School of Engineering & Technology, Amity University, Sec-125, Noida,U.P., 201301,India
skdubey1@amity.edu

Ajay Rana[3]

Department of Computer Science & Engineering, Amity School of Engineering & Technology, Amity University, Sec-125, Noida,U.P., 201301,India
ajay_rana@amity.edu

Abstract

The interest in software system is increasing day by day. Dealing with the software systems is a complex task. Software must have some qualities on the basis of which it can be applied to any software system. Every software quality model has some characteristics and sub-characteristics, which affect software quality. In this Paper the main emphasis is given on maintainability characteristics. Every system requires that maintainability measure should be done in early stages of development life-cycle which will help the designers to correct the software, if there is any fault, in early phases of designing. By doing this, cost of the development of software will be reduced. The paper gives the overview of various quality models in which maintainability is described. Paper provides the analysis of maintainability in various quality models.

Keywords**:** ***software quality; quality characteristics; quality models; maintainability.***

1. INTRODUCTION

Software engineering is a complex problem solving activity with quality goals [1].The Quality of the software can be measured by using software quality characteristics. Various software quality models having the quality characteristics are given by various researchers in the last few decades. Some of the quality characteristics are difficult to measure and therefore they receive little attention. One example of such type is quality characteristics known as maintainability, defined as the capability of the software product to be modified [2].

Software maintainability is defined as “the ease with which a software system or component can be modified to correct faults, improve performance or other characteristics, or adapt to a changed environment” [3]. Thus, the software product that is maintainable should be well-documented, should not be complex, and should have spare capacity for memory, storage and processor utilization and other resources.

The maintainability of software is probably the element that can best be approached at the level of the software's design and actual code. When talking about the software's maintainability we are interested about analyzability, how easy it is for us to locate the elements we want to improve or fix; changeability, how much work we need to do to implement a modification; stability, how few things break after our changes; and testability, our ability to validate our modifications. We deal with all these elements.

In other words, the ability to identify and fix a fault within a software component is what the maintainability addresses. Anything that helps with identifying the cause of a fault and then fixing the fault is the concern of maintainability.

Despite the fact that software maintenance is an expensive and challenging task, it is not properly managed and often ignored. One reason for this poor management is the lack of proven measures for software maintainability [4].

Maintainability is defined as the capability of the software product to be modified [5]. Modifications may include corrections, improvements or adaptations of the software to changes in the environment and in the requirements and functional specifications (the effort needed to be modified).

In software engineering, maintainability is required for correction of defects, meeting new requirements, to cope with a changing environment, and modifying a software product with ease. So we can say, in order to identify and fix a fault within software, maintainability is required.

2. SOFTWARE QUALITY MODELS

. In the last two decades researchers have proposed various quality models based on software quality. This section includes study of these models and characteristics namely reliability, usability, maintainability, efficiency, portability of a software system.

In 1977, Jim McCall proposed a software quality model called as McCall's model [6]. In this model McCall explained the 1main properties of characterizing the quality characteristics of a software product in terms of product revision, product transition and product operations. As product revision means ability to change, he identifies the quality characteristics that influence the ability to change the software product. The 11 quality characteristics are maintainability, flexibility, testability, correctness, reliability, efficiency, integrity, usability, portability, reusability, and interoperability.

Another software quality model called as Boehm's quality model was given by Barry W. Boehm in 1978. In his model he explained the various software requirements as utility, maintainability, portability, reliability, usability, efficiency. He explained Maintainability as an ease of identifying what need to be changed [7].

Robert Grady at Hewlett Packard proposed a model called as FURPS+ Model in 1987 [8]. FURPS stand for Functionality, Usability, Reliability, Performance and Supportability. He divided various characteristics into functional requirements and non-functional requirements [9]. Functional requirements explain what has to be done by identifying the necessary task, action or activity that must be accomplished. Functional requirements analysis will be used as the top level functions for functional analysis. Non-functional requirements are requirements that specify criteria that can be used to judge the operation of a system, rather than specific behaviors. He explained non-functional requirements of the software quality as functionality, usability, reliability, performance and supportability. Maintainability along with other requirements such as testability, adaptability, compatibility, scalability, localizability, and installability is explained under supportability.

Glib in 1988, proposes four qualities characteristics: workability, availability, adaptability and usability, accompanied by the resource characteristics of time, money, people and tools. He proposed these models in Principles of Software Engineering Management [10].

COQUAMO was based on the ideas proposed by Kitchenham and Pickard [11]. In 1989, Kitchenham et al. made a quantitative approach to monitoring software development. They recommend a number of factors such reliability, usability, maintainability etc. in their study.

Ghezzi et al. in 1991 state that in general, users of the software only care about the external qualities, but it is the internal qualities - which deal largely with the structure of the
software - that help developers achieve external qualities. The external quality characteristics proposed by Ghezzi are integrity, reliability, usability, accuracy, and internal quality factors include efficiency, maintainability, flexibility, re-usability, portability [12].

Another model similar to previous models is Dromey's Quality Model presented by R. Geoff Dromey in 1995. The main characteristics described by this model are Functionality, Usability, Reliability, Efficiency, Maintainability, Portability, Reusability, Process maturity [13].

ISO 9126 is an international standard for the evaluation of software. This standard is divided into four parts and the first part is related to quality model, therefore referred to as ISO 9126-1 software quality model [14]. It is an extension of the previous models such as McCall, Boehm, FURPS and others as all these defines a set of software quality characteristics. Functionality, Reliability, Usability, Efficiency, Maintainability, Portability are the six main characteristics identified by the ISO 9126-1 software quality model (ISO 9126-1).

Georgiadou et al. developed a generic, customizable quality model (GEQUAMO**)** which enables any stakeholder to construct their own model depending on their requirements. In a further attempt to differentiate between stakeholders, Siaka et al studied the viewpoints of users, sponsors and developers as three important constituencies/stakeholders and suggested characteristics of interest to each constituency as well as level of interest. More recently, Siaka and Georgiadou reported the results of a survey amongst practitioners on the importance placed on product quality characteristics. Using their empirical results they extended ISO 9126 by adding two new characteristics namely extensibility and security which have gained in importance in today's global and inter-connected environment [15]

In 2009, Ranbireshwar et. al. in their work proposed eight quality factors. These are Performance, Scalability, Cost/Benefit, Usability, Portability, Robustness, Correctness, and Reliability [16].

3. SOFTWARE QUALITY MODELS BASED ON MAINTAINABILITY

From last two decades, a wide range of maintainability prediction models have been proposed in the literature survey. Some of the models have predicted maintainability using the metrics from coding as well as design phase, while some are focusing only on design level metrics [17]. The models based on maintainability are explained below.

It has been observed that various researchers proposed several models for maintainability estimation, but in most of these studies, maintainability estimation depends on the measures taken after the coding phase. Because of this reason, maintainability predictions are made in the latter stages and it become very difficult to improve the maintainability at that stage.

In 1977, Jim McCall proposed a software quality model called as McCall's model. In this model McCall In this model McCall identified the 11 quality factors broken down by the three main perspectives for characterizing the quality attributes of a software product namely product revision (ability to change), product transition (adaptability to new environments) and product operations (basic operational characteristics. For each quality factor McCall defined one or more quality criteria (a way of measurement), in this way an overall quality assessment could be made of a given software product by evaluating the criteria for each factor. For example the maintainability quality factor would have criteria of simplicity, conciseness, and modularity as the sub characteristics.

Another software quality model called as Boehm's quality model was given by Barry W. Boehm in 1978. He defined a hierarchical model of software quality characteristics, in trying to qualitatively define software quality as a set of attributes and metrics (measurements). At the highest level of his model, Boehm defined three primary uses namely, as-is utility, the extent to which the as-is software can be used (i.e. ease of use, reliability and efficiency), maintainability, ease of identifying what needs to be changed as well as ease of modification and retesting and portability, ease of changing
software to accommodate a new environment. He explained maintainability as an ease of identifying what need to be changed. He defined testability, understandability and flexibility as the sub-characteristics of maintainability.

A *Fuzzy model* was proposed by Sneed and Mercy, which considered design characteristics [18]. In this model the maintainability is a measure of characteristics of software, eg. Source code readability, documentation quality and cohesiveness among source code and documents. They proposed a model which integrates the three characteristics namely Average number of variables, the Average Cyclomatic Complexity *(ACC*) and the Comments Ratio (*CR*) and provides a measure of maintainability is proposed. According to this model the lower the comment ratio, the better is the readability, thus better is the maintainability. So this model includes readability, documentation quality and cohesiveness as the sub characteristics of maintainability.

Robert Grady at Hewlett Packard proposed a model called as FURPS+ Model in 1987. The F in the FURPS+ acronym represents all the system-wide functional requirements that we would expect to see described. U represents Usability which includes looking at, capturing, and stating requirements based around user interface issues, things such as accessibility, interface aesthetics, and consistency within the user interface. R represents Reliability includes aspects such as availability, accuracy, and recoverability, for example, computations, or recoverability of the system from shut-down failure. P represents Performance involves things such as throughput of information through the system, system response time (which also relates to usability), recovery time, and startup time. S represents Finally, we tend to include a section called Supportability, where we specify a number of other requirements such as testability, adaptability, maintainability, compatibility, configurability, installability, scalability, localizability, and so on. The "+" of the FURPS+ acronym allow us to specify constraints, including design, implementation, interface, and physical constraints.

ISO9126-1 represents the latest research into characterizing software for the purposes of software quality control, software quality assurance and software process improvement (SPI). The ISO 9126-1 software quality model identifies 6 main quality characteristics which were further broken into sub-characteristics. The main sub-characteristics of maintainability are changeability, stability, testability and adaptability as defined by ISO 9126-1 software quality model (ISO 9126-1). It is at the sub-characteristic level that measurement for SPI will occur.

Another model similar to previous models is Dromey's Quality Model. Dromey included Self-descriptiveness, Modifiability, Testability as the sub characteristics of Maintainability.

IEEE has also release several standards, one related to the topic covered within this technical paper, IEEE STD 730-1998[19]. It is IEEE Standard for Software Quality Assurance Plans. It is activity based quality model IEEE 1219 std. Testability, implementation, and process delivery are the sub characteristics of maintainability as defined by IEEE std.

In another study Genero et al. [20]., developed four models which relate the size and structural complexity metrics with maintainability measures like understandability time, modifiability correctness and modifiability completeness. So in their models they explained understandability, modifiability correctness and modifiability completeness as the sub characteristics of maintainability. But none of the four models quantify the maintainability of class diagrams itself

In another study Kiewkanya et al.[21], proposed a maintainability model developed using weighted-sum method. The model adds the weighted values of understandability and modifiability of a class diagram, to get the corresponding maintainability value Two sub-characteristics of maintainability: understandability and modifiability are focused in this work So according to kiewkanya et al, the values of understandability and modifiability measured after the analysis are helpful in the maintainability.

According to Rizvi et al.[22], before establishing the model for maintainability, it is important to ensure the proper correlation among maintainability, understandability and modifiability of class diagrams. They find that both understandability and modifiability are strongly correlated with maintainability. While the correlation between understandability and modifiability is not so strong. This supports their candidature to be act as independent variables in the maintainability model. Measuring software maintainability early in the development life cycle, especially at the design phase, may help designers to incorporate required enhancement and corrections or improving maintainability of the final software. 'Maintainability Estimation Model for Object-Oriented software in Design phase' (MEMOOD), estimates the maintainability of class diagrams in terms of their understandability and modifiability. (MEMOOD), taking understandability and modifiability as independent, while maintainability as dependent variable.

4. ANALYSIS AND COMPARISON

Various software quality models are studied. After studying these models a comparison table is made which shows the various models and their characteristics (Table-I). Every model have some characteristics like Portability, Usability, Modifiability, Maintainability, etc. as marked in the Table-1 But here the main emphasis is given on maintainability characteristics of software quality models because it is the factor which affect the software system the most. If we can predict the maintainability at the early stages of the software development, the cost of the software can be reduced. Maintainability characteristics mean the product's ability to be changeable, maintainable, and updateable. Various software quality models, like Dromey's, McCall's, FURPS+ etc, are available in which maintainability is defined. Maintainability is evaluated by various researchers through several software quality sub characteristics such as analyzability, changeability, stability and testability.

TABLE I: CHARACTERISTIC COMPARISON

Source → Characteristics	McCall [6]	Boehm [7]	FURPS [8]	Glib [10]	Kitchenham et al [11]	Ghezzi et al.[12]	Dromey [13]	ISO-9126 [14]	Georgiadou et al. [15]	Jamwal et al. [16]
Functionality			X				X	X	X	
Correctness	X									X
Reliability	X	X	X		X	X	X	X	X	X
Usability	X	X	X	X	X	X	X	X	X	X
Availability				X						
Integrity	X					X				
Maintainability	X	X	X		X	X	X	X	X	
Reusability	X					X	X	X		
Flexibility	X					X				
Adaptability				X						
Efficiency	X	X				X	X	X	X	
Utility		X								
Extensibility									X	
Performance			X							X
Security									X	
Maturity							X	X		
Supportability			X							
Portability	X	X				X	X	X	X	X
Workability				X						
Cost/Benefit										X
Robustness										X
Accuracy						X				

Another comparison table is there (Table-II) on the basis of sub characteristic of maintainability. Here we analyze those sub characteristic which affects the maintainability the most.

TABLE II: MAINTAINABILITY SUB- CHARACTERISTIC COMPARISON

Source → Sub-characteristics	McCall [6]	Boehm [7]	Fuzzy Model [18]	FURPS [8]	ISO 9126 [14]	Dromey [13]	IEEE [9]	Genero et al [20]	Kiewkanya et al [21]	Rizvi et al.[22]
Changeability	X	X		X	X					
Readability			X							
Stability					X					
Documentation			X							
Simplicity	X									
Conciseness	X									
Cohesiveness			X							X
Self-descriptiveness	X					X		X		
Modifiability		X				X		X	X	X
Testability	X	X		X	X	X	X			
Adaptability				X	X					
Flexibility		X								
Implementation							X			
Delivery							X			
Compatibility				X						
Localizability				X						
Install ability				X						

A bar chart (Figure-1) explains about the frequency of sub-characteristics of maintainability. It helps to determine those sub-characteristics, the repeated occurrence of which sub-characteristics affects the maintainability of the software system the most.

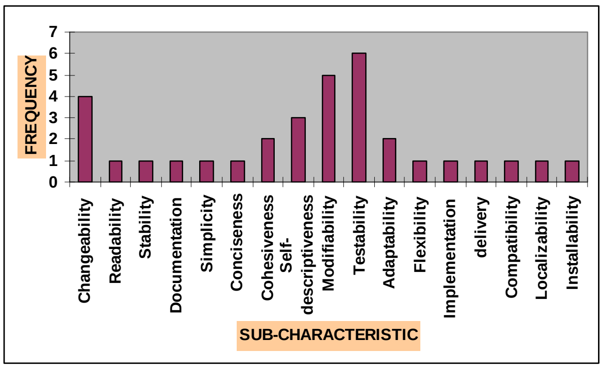

FIGURE1: Maintainability Sub-Characteristic Chart

5. CONCLUSION

The present work involves detailed study of various software quality models in order to quantify the factors which affect software quality. There are various characteristic and sub-characteristic in a software quality system that affects software performance. One of the important factor which is of immediate use in the software development process is maintainability. Maintainability is the ease with which a product can be maintained in order to:

- correct defects,
- meet new requirements,
- make future maintenance easier, or
- cope with a changed environment.

The software maintainability is an important characteristic of a software model which is intended to help reduce a system's tendency, and to indicate when it becomes cheaper and less risky to rewrite the code instead to change it. The proper knowledge of maintainability enables the maintenance team to know, on what module to focus during maintenance.

In the future importance will be given on the sub characteristics which affects the maintainability. If these sub characteristics are analysed correctly then maintenance of the software will be easier. If object oriented mechanism along with the maintainability are used conveniently in the early stages of software lifecycle. They will help to reduce maintainability effort. We are going to propose model based on these sub-characteristics .and more sophisticated maintainability estimation model can also be developed in future, by conducting a larger scale study with a variety of industrial projects across diverse domains.

References

[1] Boehm,B et al. In(1996) "Identifying Quality-Requirement Conflicts," *IEEE Software*, vol. 13, pp. 25-35.

[2] ISO/IEC, "Software engineering — Product quality — Part 1: Quality model," 2001.

[3] IEEE Std. 610.12-1990. Standard Glossary of Software Engineering Terminology, IEEE Computer Society Press, Los Alamitos, CA, 1993.

[4] Dagpinar, M., Jahnke, J.,(2003) "Predicting Maintainability with Object- Oriented Metrics – an Empirical Comparison," Proc. 10th Working Conference on Reverse Engineering (WCRE'03), 13 - 17 Nov. 2003, pp. 155 -164.

[5] International Organization for Standardization. ISO/IEC 9126-1: Software engineering - Product quality - Part 1: Quality model, 2001.

[6] McCall, J.A., Richards, P.K., and Walters, G.F.,(1977) "Factors in Software Quality", RADC TR-77-369, Vols I, II, III, US Rome Air Development Center Reports.

[7] Boehm, B. W., Brown, J. R., Kaspar, H., Lipow, M., McLeod, G., and Merritt, M.,(1978) *Characteristics of Software Quality*, North Holland.

[8] Grady, Robert, Caswell, Deborah(1987), *Software Metrics: Establishing a Company-wide Program*. Prentice Hall. pp. p. 159. ISBN 0138218447.

[9] http:// en.wikipedia.org/wiki/ Requirements_ analysis # Types_of_Requirements.

[10] Gilb, T.,(1988) *Principles of Software Engineering Management*, Addison-Wesley, Reading MA.

[11] Kitchenham, B., Pickard, L.,(1989) Towards a constructive quality model. Part 2: Statistical techniques for modelling software quality in the ESPRIT REQUEST project, City University, Centre for Software Reliability, London, UK , Software Engineering Journal,pp 114-126.

[12] Ghezzi, C., Jazayeri, M., Mandrioli, D.(1991), Fundamentals of Software Engineering. Prentice-Hall.

[13] Geoff R. Dromey's Model, (Feb 1995) (vol. 21 no. 2),IEEE Transaction On Software Engineering, A Model for Software Product Quality.

[14] ISO 9126-1 Software Engineering - Product Quality - Part 1: Quality Model, 2001.

[15] Georgiadou, E.,(November 2003) "GEQUAMO– A Generic, Multilayered, Customisable, Software Quality Model", International Journal of Cybernetics, Volume 11, Number 4 , pp 313-323.

[16] Ranbireshwar S. Jamwal & Deepshikha Jamwal(2009), Issues & Factors For Evaluation of Software Quality Models, Proceedings of the 3rd National Conference; INDIACom.

[17] Rizvi, S.W.A., Khan, R.A.(2009), "A Critical Review on Software Maintainability Models," *Proc. of the National Conference on Cutting EdgeComputer and Electronics Technologies*, 14 - 16 Feb. 2009, pp. 144 – 148, Pantnagar, India.

[18] Sneed, H., Mercy, A. (1985), Automated Software Quality Assurance. IEEE Trans. Software Eng., 11Bi,9: 909-916.

[19] IEEE Standard for Software Quality Assurance Plans, pp. 730-1998.

[20] Genero, M., Manso, E., and Cantone, G.(2003), "Building UML Class Diagram Maintainability Prediction Models Based on Early Metrics," *Proc. 9th International Symposium on Software Metrics (METRICS'03)*, 3 -5 Sept., 2003, pp. 263 – 275.

[21] Kiewkanya, M., Jindasawat, N., Muenchaisri, P.,(2004) "A Methodology for Constructing Maintainability Model of Object-Oriented Design," *Proc. 4th International Conference on Quality Software*, 8 - 9 Sept., 2004, pp. 206 - 213. IEEE Computer Society.

[22] Rizvi, S.W.A., Khan, R.A.(April 2010), Maintainability Estimation Model for Object-Oriented Software in Design Phase, Journal of Computing, Volume 2, Issue 4.